

2

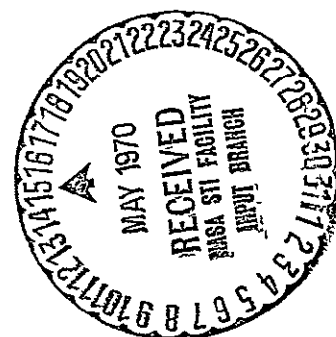
The University of Arizona
Engineering College
Department of Electrical Engineering
Computer Science Research Laboratory

CSRL Memo No. 199

LOCUST: A TEST PROGRAM
FOR THE PDP-9/LOCUST HYBRID INTERFACE

January, 1970

by W. R. Moore



ACKNOWLEDGEMENT:

The writer is grateful to the National Aeronautics and Space Administration for their support of this work under NASA Grant ~~NSG-644~~. *NG-03-002-024*

FACILITY FORM 002	N 70-27171	
	(ACCESSION NUMBER)	(THRU)
	<i>32</i>	<i>1</i>
	(PAGES)	(CODE)
	CR-109843	08
	(NASA CR OR TMX OR AD NUMBER)	(CATEGORY)

Reproduced by the
CLEARINGHOUSE
for Federal Scientific & Technical
Information Springfield Va 22151

INTRODUCTION

LOCUT1 is a Fortran-IV main program which calls Macro-9 subroutines to test the Computer Science Research Laboratory's hybrid computer interface between the FDP-9 digital computer and the LOCUST high speed iterative analog computer. The programs are written to provide the user with all the information he needs to perform the tests, assuming of course a reasonable familiarity with both of the computers and the interface itself. The programs ask questions, give patching and test operating instructions, and print out diagnostic messages on the teletype to point out any malfunction in the interface.

In writing the programs, priority was given to user convenience, not to programming efficiency. Because of this and the secondary purpose of the project, to familiarize the writer with FDP-9 programming, a variety of things were done "the hard way."

LOCUT1 in its present form performs four basic tests of the interface by calling four subroutines: the free flag/IOT test by calling FFIOT; the control register/read-in gate test by calling CONREG; the pre-set counters by calling PRECTR; and the MDAC/ADC test by calling CONV.

FREE FLAG/IOT TEST

FFIOT is the subroutine called by LOCUT1 to test the free flags and patchable IOT's that appear on the LOCUST digital patchbay. As in all of the interface tests, one interface component must be used to test another; here, the IOT's are issued by the PDP-9 and then used to set the free flags. Since the free flags remain set only while there is a logical "1" input to them, the IOT's are run through ~~an~~ a toggled J-K flip flop to set the flags. This means that after a flag has been checked with a skip instruction, it must be cleared by issuing the IOT again to toggle the flip flop to a "0." Care must also be taken to insure that the flip flop is initially reset to a "0" output when the test begins. This is best done by using one of the logic state indicator lamps and a push button connected to the reset input of the flip flop.

FFIOT initializes six locations to contain one. If one of the flags fails to set when one of the IOT's is issued, one of these six locations is set to zero. If the test is successful, the location is not changed. When control returns to LOCUT1, these locations are converted to logical variables and tested to determine which diagnostic messages to print out. These logical tests are necessary since a negative result from one free flag/IOT pair is not sufficient for a conclusion; it must be compared with a positive result which includes one of the pair.

CONTROL REGISTER/READ-IN GATE TEST

The 6-bit control register is tested in conjunction with the the 6-bit read-in gates. The control register outputs are patched to the read-in gate inputs bit for bit and to six logic state indicator lamps which are used as a visual check in case the read-in gates fail. This visual check requires some effort on the part of the user, but much less effort than would be involved in patching a logic scheme that would avoid it. LOCUT1 calls CONREG which first transfers all ones then all zeros to the control register and checks the result fed back by the read-in gates. If any bits fail, a macro is called which determines the error bits and instructs the user in further tests. If the control register is at fault, one of patchable IOT's must be used to check the read-in gates. CONREG uses Macro-9 input-output statements to provide the user with documentation of the test results.

Note that the user must be sure that the read-in gates are properly patched to accumulator bits 0-5 in the data channel patchbay, and that the read request pulse is provided there.

PRE-SET COUNTER TEST

The pre-set counters are checked with PRECTR and ERROR, a Fortran subroutine called by PRECTR. The decimal number 600 is loaded into each counter and then IOT's are counted

until 600 of them have been issued or until the counter "s" output goes through its positive logical transition. If the counter fails to set after 600 pulses, the program informs the user that that is the case. If the counter sets before 600 pulses have been issued, PRECTR calls ERROR which prints out the number which the counter did count before it set.

In this test toggled J-K flip flops are also used and care must be used in insuring that they are initially reset.

MDAC/ADC TEST

The MDAC/ADC test differs from the other three in that the user is not only interested in whether the components work, but in how accurate they are. Therefore the program allows a great deal of flexibility. The user selects any MDAC/ADC pair and patches them as instructed. A digital quantity is set in the accumulator switches 0-10 and read into the MDAC by CONV. By patching the MDAC output to an ADC input, and to a digital voltmeter, both the MDAC and ADC accuracy can be checked: the MDAC by comparing the accumulator switches with the DVM reading, and the ADC by comparing the DVM reading with the quantity remaining in the accumulator ~~switches~~ when the program halts. To check another quantity, the new value is set in the accumulator and the "CONTINUE" switch on the PDP-9 keyboard is depressed. To check another MDAC/ADC pair, octal 000001 is set in the

accumulator switches and the "CONTINUE" switch again depressed. The program then asks for the number of the MDAC and the ADC and proceeds as before. To terminate the test, enter zero for the MDAC number.

OPERATING NOTES

The program allows the user to request patching instructions, or ^{to}/have them ~~be~~ suppressed. Any tests the user does not wish to run may also be suppressed. However, unless the user is certain that the control register and the free flags and IOT's are properly operating, they should be tested before running the pre-set counter test or the MDAC/ADC test. PRECTR uses the free flags and IOT's, and CONV uses the control register.

If all tests are to be run, digital patchbay economy can be achieved by patching the free flag/IOT test and the control register/ read-in gate test on the same patchbay, and the pre-set counter and MDAC/ADC test on the same patchbay. Analog patching is required only for the converter tests, and this is the only time LOCUST must be in any mode other than patch.

CONCLUSION

It is felt that programs of this type will be a great help in using and maintaining the PDP-9 LOCUST interface.

LOCUT1 can easily be expanded to test other interface functions as they are added, e.g. the data channel. As the system now stands, the only function not exercised by LOCUT1 is the program interrupt facility. Future versions of the program will be named LOCUT2, LOCUT3, etc.

The programs are written to be self-explanatory, but the user must be thoroughly familiar with LOCUST, the PDP-9 and the interface. The programs themselves and two sample test print outs are included at the end of this paper. the first print out includes all patching instructions and tests and indicates a properly working interface. The second one is an example of the diagnostics output when there are malfunctions.

REFERENCES

1. Conant, Brian K., Design of a New Solid State Electronic Differential Annalyzer Making Use of Integrated Circuits, Ph. D. Dissertation, University of Arizona, 1968.
2. Goltz, John R., An Economical Analog-to-Digital Converter System for Hybrid Computers, ACL Memo No. 147.
3. Wilkins, Jeff, A Modern Hybrid Computer Interface, Masters Thesis, University of Arizona, 1969.

MONITOR VAB

SLOAD

LOADER V5A

>LOCUTI, FFIOT, CONREG, PRECTR, ERROR, CONV

LOCUTI 33650
FFIOT 33512
CONREG 31167
PRECTR 30723
ERROR 30645
CONV 30473
.DA 30424
BCDIO3 25263
.SS 25204
FIOPS 24450
OTSER 24354
INTEGE 24236
REAL 23233
↑S↑Q↑

MONITOR VAB

SG 1
↑S

"LOCUT1"
PDP-9/LOCUST HYBRID INTERFACE TEST!

DO YOU WISH PATCHING INSTRUCTIONS?

YES

TYPE "NO" AFTER TESTS NOT TO BE RUN.

FREE FLAG/IOT TEST?

YES

CONTROL REGISTER/READ-IN GATE TEST?

YES

PRE-SET COUNTER TEST?

YES

MDAC/ADC TEST?

YES

FREE-FLAG/IOT TEST:

PATCH EACH "PATCHABLE" IOT TO A 3-INPUT "OR"
GATE AND THE GATE OUTPUT TO J&K INPUTS
OF A FLIP FLOP. THEN PATCH THE FLIP
FLOP OUTPUT TO BOTH FREE FLAGS. (NOTE
THAT THE FLIP FLOP IS TOGGLED AND MUST
BE RESET BEFORE STARTING THE TEST.)

TYPE "GO" WHEN PATCHING COMPLETE.

GO

FREE FLAG 0 OKAY

FREE FLAG 1 OKAY

IOT1 OKAY

IOT2 OKAY

IOT4 OKAY

FREE-FLAG/IOT TEST COMPLETE!

CONTROL REGISTER/READ-IN GATE TEST:

PATCH CONTROL REGISTER OUTPUTS TO
CORRESPONDING READ-IN GATE
INPUTS AND TO 6 LOGIC-STATE
INDICATOR LAMPS. THE LAMPS ARE
USED AS A VISUAL CHECK IN CASE
THE READ-IN GATES FAIL.

TYPE "GO" WHEN PATCHING COMPLETE.

GO

CONTROL REGISTER & READ-IN#GATES OKAY.

CONTROL REGISTER/READ-IN GATE TEST
COMPLETE!

PATCHING FOR PRESET COUNTER TEST:
PATCH IOT1 TO COUNTER 1, COUNTER 1'S
S OUTPUT TO J&K INPUTS OF A6FLIP FLOP,
AND THE FLIP FLOP OUTPUT TOCFREE FLAG 0.
PATCH IOT2 TO COUNTER 2, COUNTER 2'S
S OUTPUT TO J&K INPUTS OF ACFLIP FLOP,
AND THE FLIP FLOP OUTPUT TOCFREE FLAG 1.
PATCH IOT4 TO THE RESET INPUTS
OF BOTH FLIP FLOPS.
FLIP FLOPS MUST BE INITIALLY RESET!!

TYPE "GO" WHEN PATCHING COMPLETE.

GO

PRE-SET COUNTER 1 OKAY.
PRE-SET COUNTER 2 OKAY.

PRESET COUNTER TEST COMPLETE!

ADC/MDAC TEST:
PATCH CONTROL REGISTER BIT 0 TO A
NOR GATE, THE NOR OUTPUT OF3WHICH
GOES TO EXT. IR AND THE OR OUTPUT
GOES TO EXT. CP.
PATCH CONTROL REGISTER BITS 1-4 EACH
TO A NOR GATE WITH THE COMPLEMENT
OF THE R PULSE. THEN PATCH
NOR GATE #1 TO A/D 1, ETC., SO THAT
THE ADC CONTROL PULSE "R" APPEARS
ONLY WHEN THE CORRESPONDING CONTROL
REGISTER BIT IS A "0".

ANALOG PATCHING:
PATCH EACH MDAC TO AN ADC INPUT AND
TO THE DVM. TYPE MDAC # AND ADC #
OF EACH PAIR TO BE TESTED (ENTER
NOTHING TO TERMINATE TEST).

TEST OPERATION: SET AC SWITCHES 0-10
FOR MDAC READ-IN. WHEN PROGRAM
HALTS, ADC READS INTO AC.
TO CHANGE INPUT, SET AC 0-10 AGAIN
AND DEPRESS "CONTINUE".
TO CHANGE ADC/MDAC PAIR, PUT UP AC 17

NDAC# :

3

ADC# :

1

TYPE "GO" WHEN PATCHING COMPLETE:

G

NDAC# :

2

ADC# :

2

TYPE "GO" WHEN PATCHING COMPLETE.

G

NDAC# :

4

ADC# :

3

TYPE "GO" WHEN PATCHING COMPLETE.

G

NDAC# :

ALL TESTS COMPLETE!

TYPE GO TO RESTART PROGRAM.

"LOCUT1"
PDP-9/LOCUST HYBRID INTERFACE TEST1

DO YOU WISH PATCHING INSTRUCTIONS?

N

TYPE "NO" AFTER TESTS NOT TO BE RUN.

FREE FLAG/IOT TEST?

CONTROL REGISTER/READ-IN GATE TEST?

PRE-SET COUNTER TEST?

MDAC/ADC TEST?

N

TYPE "GO" WHEN PATCHING COMPLETE.

G

IOT1 OKAY
FREE FLAG 0 BAD
FREE FLAG 1 OKAY
IOT2 BAD
IOT4 OKAY

FREE-FLAG/IOT TEST COMPLETE!

CONTROL REGISTER/READ-IN GATE TEST -
TYPE "GO" WHEN PATCHING COMPLETE.

G

A PROBLEM EXISTS WITH THE FOLLOWING BITS
IN EITHER CONTROL REGISTER OR READ-IN
GATES. AFTER EACH PRINT OUT CHECK THE
LAMP CORRESPONDING TO THAT BIT. IF IT IS
LIT TYPE AN R INDICATING THAT THE READ-
IN GATE IS IN ERROR. IF LAMP NOT LIT
TYPE C INDICATING ERROR IN CONTROL
REGISTER AND POSSIBLE ERROR IN
READ-IN GATE.

ERROR IN BIT 1
TYPE R OR C - R

READ-IN GATE ERROR IN ABOVE BIT.

ERROR IN BIT 3
TYPE R OR C -

C

CONTROL REGISTER ERROR AND POSSIBLY
READ-IN GATE ERROR IN ABOVE BIT.
PATCH A WORKING IOT TO J&K INPUTS OF A
FLIP FLOP AND ITS OUTPUT TO THE ERROR
BIT IN THE READ-IN GATE, THEN TYPE G.
G

CONTROL REGISTER & READ-IN GATE ERROR.

NO FUTHUR ERRORS.

CONTROL REGISTER/READ-IN GATE TEST
COMPLETE!

PRESET COUNTER TEST -
TYPE "GO" WHEN PATCHING COMPLETE.

G

PRE-SET COUNTER 1 OKAY.
PRE-SET COUNTER 2 OKAY.

PRESET COUNTER TEST COMPLETE!

ALL TESTS COMPLETE!
TYPE GO TO RERUN PROGRAM.

TYPE CTRL S TO RESTART.

C A FORTRAN MAIN PROGRAM WHICH CALLS
 C MACRO SUBROUTINES TO TEST THE
 C LOCUST/PDP-9 HYBRID INTERFACE.
 C COMMUNICATION WITH USER IS VIA
 C TELETYPE, BUT LINE LENGTHS
 C CONFORM TO CRT EDITING.

C

C WM. R. MOORE JANUARY, 1970

C

INTEGER TESTF0(3), TESTFI(3), ADC
 LOGICAL FI(6), PATCH
 REAL NO

DATA YES/1NY/, NO/1NN/, GO/1HG/

PATCH = .TRUE.

WRITE(5,2222)

2222 FORMAT(1H1,16X,* "LOCUT1"*/2X,
 ** PDP-9/LOCUST HYBRID INTERFACE TE
 #STI*///)

WRITE(5,1)

1 FORMAT(* DO YOU WISH PATCHING INS
 #TRUCTIONS? *)

READ(4,2) ANS1

2 FORMAT(A1)

IF (ANS1.EQ.NO) PATCH = .FALSE.

WRITE(5,10)

10 FORMAT(* TYPE "NO" AFTER TESTS N
 #OT TO BE RUN.*//)

WRITE(5,11)

11 FORMAT(* FREE FLAG/IOT TEST?*)

READ(4,2) ANS2

WRITE(5,12)

12 FORMAT(* CONTROL REGISTER/READ-I
 #N GATE TEST?*)

READ(4,2) ANS3

WRITE(5,13)

13 FORMAT(* PRE-SET COUNTER TEST?*)

READ(4,2) ANS4

WRITE(5,14)

14 FORMAT(* MDAC/ADC TEST?*)

READ(4,2) ANS5

IF(ANS2.EQ.NO) GO TO 15

IF(.NOT.PATCH) GO TO 50

WRITE(5,100)

100 FORMAT(* FREE-FLAG/IOT TEST: */
 #1X,* PATCH EACH "PATCHABLE" IOT TO
 # A 3-INPUT "OR"*/1X,* GATE AND THE
 # GATE OUTPUT TO J&K INPUTS*/1X,
 ** OF A FLIP FLOP, THEN PATCH THE F

```

#LIP*/IX,* FLOP OUTPUT TO BOTH FREE
# FLAGS. (NOTE */IX,* THAT THE FLIP
# FLOP IS TOGGLED AND MUST*/IX,* BE
# RESET BEFORE STARTING THE TEST.)*
#//)
50  WRITE(5,51)
51  FORMAT(* TYPE "GO" WHEN PATCHING
# COMPLETE.*//)
54  READ(4,2) ANSI
    IF (ANSI.EQ.GO) GO TO 55
    GO TO 50
55  WRITE(5,56)
56  FORMAT(1H0)
    CALL FFIOT (TESTF0(1),TESTF0(2),
# TESTF0(3),TESTF1(1),TESTF1(2),
# TESTF1(3))
    DO 3 I=1,3
3    FI(1)=TESTF0(1).EQ.1
    DO 4 J=4,6
    K=J-3
4    FI(J)=TESTF1(K).EQ.1
    IF (FI(1).AND.FI(4)) GO TO 200
    IF (FI(2).AND.FI(5)) #GO TO 300
    IF (FI(3).AND.FI(6)) #GO TO 400
    IF (FI(1).OR.FI(4)) #GO TO 500
    IF (FI(2).OR.FI(5)) GO TO 600
    IF (FI(3).OR.FI(6)) GO TO 700
    WRITE (5,800)
800  FORMAT(* FREE-FLAG/IOT TEST FAILS
#,* / * ONE OF THE FOLLOWING IS TRU
#E:* / * (1) PATCHABLE IOTS ARE INO
#PERATIVE;* / * (2) BOTH FREE-FLAGS
# ARE INOPERATIVE;* / * (3) BOTH (1
#) AND (2).*//)
    GO TO 900
200  WRITE(5,801)
    WRITE(5,802)
    WRITE(5,803)
    IF (FI(1).AND.FI(2)) GO TO 201
    IF (FI(1).AND.FI(3)) GO TO 202
    WRITE(5,814)
204  WRITE(5,815)
    GO TO 900
201  WRITE(5,804)
    IF (FI(1).AND.FI(3)) #GO TO 203
    GO TO 204
203  WRITE(5,805)
    GO TO 900

```



```
202  WRITE(5,814)
      WRITE(5,805)
      GO TO 900
300  WRITE(5,801)
      WRITE(5,802)
      WRITE(5,813)
      WRITE(5,804)
      IF (FI(2).AND.FI(3)) GO TO 203
      GO TO 204
400  WRITE(5,801)
      WRITE(5,802)
      WRITE(5,813)
      WRITE(5,814)
      GO TO 203
500  WRITE(5,803)
      IF (FI(1)) GO TO 501
      WRITE(5,811)
      WRITE(5,802)
      IF (FI(4).AND.FI(5)) GO TO 502
      IF (FI(4).AND.FI(6)) GO TO 505
      WRITE(5,814)
      GO TO 204
502  WRITE(5,804)
      IF (FI(1).AND.FI(3)) GO TO 203
      GO TO 204
505  WRITE(5,814)
      GO TO 203
501  WRITE(5,801)
      WRITE(5,812)
      IF (FI(1).AND.FI(2)) GO TO 502
      IF (FI(1).AND.FI(3)) GO TO 505
      WRITE(5,814)
      GO TO 204
600  WRITE(5,813)
      WRITE(5,804)
      IF (FI(2)) GO TO 601
      WRITE(5,811)
      WRITE(5,802)
      IF (FI(5).AND.FI(6)) GO TO 203
      GO TO 204
601  WRITE(5,801)
      WRITE(5,812)
      IF (FI(2).AND.FI(3)) GO TO 203
      GO TO 204
700  WRITE(5,813)
      WRITE(5,814)
      WRITE(5,805)
      IF (FI(3)) GO TO 701
```

```

        WRITE(5,811)
        WRITE(5,802)
        GO TO 900
701    WRITE(5,801)
        WRITE(5,812)
900    WRITE(5,901)
901    FORMAT(/* FREE-FLAG/IOT TEST COMP
#LETEI* //)
15    IF(ANS3.EQ.NO) GO TO 16
        IF(.NOT.PATCH) GO TO 104
        WRITE(5,101)
101    FORMAT(* CONTROL REGISTER/READ-IN
# GATE TEST:*/IX,* PATCH CONTROL RE
#GISIER OUTPUTS TO*/IX,* CORRESPOND
#ING READ-IN GATE*/IX,* INPUTS AND
# TO 6 LOGIC-STATE*/IX,* INDICATOR
# LAMPS. THE LAMPS ARE*/IX,* USED A
#S A VISUAL CHECK IN CASE*/IX,* THE
# READ-IN GATES FAIL.*//)
        GO TO 102
104    WRITE(5,105)
105    FORMAT(* CONTROL REGISTER/READ-IN
# GATE TEST -*)
102    WRITE(5,51)
        READ(4,2) ANSI
        IF(ANS1.EQ.GO) GO TO 103
        GO TO 102
105    WRITE(5,56)
        CALL CONREG
        WRITE(5,125)
125    FORMAT(/* CONTROL REGISTER/READ-I
#N GATE TEST*/3X,* COMPLETEI*//)
16    IF(ANS4.EQ.NO) GO TO 17
        IF(.NOT.PATCH) GO TO 162
        WRITE(5,150)
150    FORMAT(* PATCHING FOR PRESET COUN
#TER TEST:*/IX,* PATCH IOT1 TO COUN
#TER 1, COUNTER 1'S */IX,* S OUTPUT
# TO J&K INPUTS OF A FLIP FLOP,*/IX
#,* AND THE FLIP FLOP OUTPUT TO FRE
#E FLAG 0.*/IX,* PATCH IOT2 TO COUN
#TER 2, COUNTER 2'S */IX,* S OUTPUT
# TO J&K INPUTS OF A FLIP FLOP,*/IX
#,* AND THE FLIP FLOP OUTPUT TO FRE
#E FLAG 1.*/IX,* PATCH IOT4 TO THE
# RESET INPUTS*/IX,* OF BOTH FLIP F
#LOPS.*/3X,* FLIP FLOPS MUST BE INI
#TIALY RESET!!*//)

```

```

      GO TO 160
162  WRITE(5,163)
163  FORMAT(* PRESET COUNTER TEST ~*)
160  WRITE(5,51)
149  READ(4,2) ANSI
      IF(ANS1.EQ.60) GO TO 161
      GO TO 160
161  WRITE(5,56)
154  CALL PRECTR
      WRITE(5,152)
152  FORMAT(/* PRESET COUNTER TEST COM
#PLETEI*//)
17  IF(ANS5.EQ.NO) GO TO 1111
      IF(.NOT.PATCH) GO TO 186
175  WRITE(5,176)
176  FORMAT(* ADC/MDAC TEST:*/IX,* PAT
#CH CONTROL REGISTER BIT 0 TO A*/IX
#,* NOR GATE, THE NOR OUTPUT OF WHI
#CH*/IX,* GOES TO EXT. IR AND THE 0
#R OUTPUT*/IX,* GOES TO EXT. CP.*/I
#X,* PATCH CONTROL REGISTER BITS 1-
#4 EACH*/IX,* TO A NOR GATE WITH TH
#E COMPLEMENT*/IX,* OF THE R PULSE.
# THEN PATCH*/IX,* NOR GATE #1 TO A
#/D 1, ETC., SO THAT*/IX,* THE ADC
# CONTROL PULSE "R" APPEARS*/IX,* 0
#NLY WHEN THE CORRESPONDING CONTROL
#*/IX,* REGISTER BIT IS A "0". */
#* ANALOG PATCHING:*/IX,
#* PATCH EACH MDAC TO AN ADC INPUT
# AND*/IX,* TO THE DVM. TYPE MDAC #
# AND ADC #*/IX,* OF EACH PAIR TO
#E TESTED (ENTER*/IX,* NOTHING TO T
#ERMINATE TEST).*)
      GO TO 185
186  WRITE(5,184)
184  FORMAT(/* MDAC/ADC ~*)
183  WRITE(5,183)
183  FORMAT(* TEST OPERATION: *,
#* SET AC SWITCHES 0-10*/IX,* CFOR M
#DAC READ-IN. WHEN PROGRAM*/IX,* HA
#LTS, ADC READS INTO AC.*/IX,* TO C
#HANGE INPUT, SET AC 0-10 AGAIN*/IX
#,* AND DEPRESS "CONTINUE."*/IX,* T
#O CHANGE ADC/MDAC PAIR, PUT UP AC
# 17*/IX,* ONLY, THEN DEPRESS CONTI
#UE.*///)
177  WRITE(5,178)

```

```
178      FORMAT(* MDAC# :*)
      READ(4,179)MDAC
      IF(MDAC.EQ.0) GO TO 1111
179      FORMAT(11)
      WRITE(5,180)
180      FORMAT(* ADC# :*)
      READ(4,179)ADC
182      WRITE(5,56)
      WRITE(5,51)
      READ(4,2) ANSI
      IF(ANSI.EQ.60) GO TO 181
      GO TO 182
181      WRITE(5,56)
      CALL CONV (MDAC,ADC)
      GO TO 177
1111     WRITE(5,1112)
1112     FORMAT(* ALL TESTS COMPLETE!*/* I
      #YPE GO TO RESTART PROGRAM.*//)
      GO TO 54
C FORMAT STATEMENTS FOR
C FREE-FLAG/IOT TEST.
801     FORMAT(* FREE FLAG 0 OKAY*)
802     FORMAT(* FREE FLAG 1 OKAY*)
803     FORMAT(* IOT1 OKAY*)
804     FORMAT(* IOT2 OKAY*)
805     FORMAT(* IOT4 OKAY*)
811     FORMAT(* FREE FLAG 0 BAD*)
812     FORMAT(* FREE FLAG 1 BAD*)
813     FORMAT(* IOT1 BAD*)
814     FORMAT(* IOT2 BAD*)
815     FORMAT(* IOT4 BAD*)
      END
```

```

/MACRO SUBROUTINE FOR LOCUT1 TO TEST
/FREE FLAGS AND PATCHABLE IOTS.
    .TITLE FFIOT
    .GLOBL FFIOT,.DA
FFIOT 0 /ENTRY POINT.
    JMS* .DA /LINK WITH LOCUT1.
    JMP .+7 /JUMP ARGUMENT LIST.
TESTF0 0 /F01 (IN F(XY), X DENOTES
0 /F02 (THE FLAG BEING TESTED,
0 /F04 (Y DENOTES THE IOT USED
TESTF1 0 /F11 (TO TEST IT.)
0 /F12
0 /F14
/
    LAC (DAC* TESTF0) /INITIALIZE
    DAC INCI /INCI.
    LAC (-6 /INITIALIZE
    DAC INCRI /INCRI.
    INCRI 0 /LOOP INDEX.
    LAC (1 /INITIALIZATION OF
    INCI 0 /THE PRECEEDING
    ISZ INCI /ARGUMENTS TO 1.
    ISZ INCRI /THIS ALLOWS ASSUMP-
    JMP INCI /TION OF A PROPERLY
    /WORKING COMPONENT,
    /UNLESS PROVEN
    /OTHERWISE.
/
/INITIALIZE TEST INSTRUCTIONS.
/
    LAC (701621
    DAC IOTFF0
    DAC TOG
    LAC (DZM* TESTF0)
    DAC IOTFFA
    LAC (DZM* TESTF1)
    DAC IOTFFB
/
/TEST BOTH FREE FLAGS (FF0 & FF1) WITH
/ALL THREE PATCHABLE IOTS.
/
    703444 /CLEAR FF0.
    703544 /CLEAR FF1.
    IOTFF0 0 /ISSUE IOT.
    701541 /SKIP ON FF0 SET.
    IOTFFA 0 /0 IF FF0 NOT SET.
    ISZ IOTFFA /SET UP FOR NEXT IOT.
    701521 /SKIP ONCFF1 SET.

```

```
IOIFFB 0      /0 IF FFI NOT SET.
TOG      0      /TOGGLE FLIP-FLOP.
      LAC IOIFF0 /TESTCFOR
      SAD (701624 /LAST IOT.
      JMP .+7
      ISZ IOIFFB /SET UP FOR NEXT IOT.
      AND (7      /CHANGE THE IOT THAT
      ADD IOIFF0 /IS ISSUED TO
      DAC IOIFF0 /701622, AND THEN
      DAC TOG      /TO 701624.
      JMP IOIFF0
      JMP* FFIOT
      .END
```

```

/
/MACRO SUBROUTINE CALLED BY LOCUT1 TO
/TEST THE CONTROL REGISTER AND READ-IN
/GATES. USES MACRO-9 I/O.
/

```

```

        .TITLE CONREG
        .NOGEN /SUPPRESS MACRO GENERATION
        .GLOBL CONREG
        .IODEV 4,5
CONREG 0          /ENTRY POINT.
        .INIT 4,0,CONREG /INPUT.
        .INIT 5,1,CONREG /OUTPUT.
        LAC (770000
701647 /CLEAR, LOAD & TRANSFER ALL
        /1'S TO CONTROL REGISTER.
702212 /TRANSFER READ-IN GATES,
        /WHICH ARE PATCHED TO
        /CONTROL REGISTER, BACK
        /TO PDP-9.
        AND (770000 /MASK LAST 12 BITS.
        SAD (770000 /TEST THE RESULT.
        JMP T1 /IF 1'S TRANSFERED OK,
        /CHECK WITH ALL 0'S.
        DAC RES /SAVE TEST RESULT.
        JMP T2 /IF 1'S DON'T TRANSFER,
        /CHECK THE BITS MANUALLY.
T1      LAC (0
701647 /CLEAR, LOAD & TRANSFER ALL
        /0'S TO CONT. REG.
702212 /TRANSFER READ-IN GATES.
        AND (770000
        SNA /TEST THE RESULT.
        JMP ALLOK /IF BOTH 0'S AND 1'S
        /TRANSFER OK, TEST COM-
        /PLETE.
        CMA /SET ERROR BITS TO 0.
        DAC RES /SAVE TEST RESULT.
        LAC (770000 /RELOAD CONTROL
701647 /REGISTER WITH 1'S.
T2      .WRITE 5,2,NOTEP,NOTEPV
        .WAIT 5
        .WRITE 5,2,CONT1,CONT1V
        .WAIT 5
        .WRITE 5,2,CONT2,CONT2V
        .WAIT 5
        .WRITE 5,2,CONT3,CONT3V
        .WAIT 5
        .WRITE 5,2,CONT4,CONT4V

```

```

.WAIT 5
.WRITE 5,2,CONT5,CONT5V
.WAIT 5
.WRITE 5,2,CONT6,CONT6V
.WAIT 5
.WRITE 5,2,CONT7,CONT7V
.WAIT 5
.WRITE 5,2,CONT8,CONT8V
.WAIT 5
.WRITE 5,2,CARR,CARRV
.WAIT 5

```

```

/
/DEFINE MACROS TO TEST BITS MANUALLY.
/

```

```

.DEFIN BIT,LIT,NOTEA,NOTEAV
.ETC A,B,C,NEXT
LAC RES
AND LIT /MASK ALL BUT ERROR BIT.
SZA
JMP NEXT /JMP OUT OF MACRO.
.WRITE 5,2,NOTEA,NOTEAV
.WAIT 5
.WRITE 5,2,NOTEX,NOTEXV
.WAIT 5
.READ 4,2,INBUF,4
.WAIT 4
.WRITE 5,2,CARR,CARRV /CARRIAGE
.WAIT 5 /RETURN.
LAC INBUF+2
AND (774000 /MASK LAST 11 BITS.
SAD (414000 /ASCII C.
JMP A /READ-IN GATE TEST.
.WRITE 5,2,NOTER,NOTERV
.WAIT 5
.WRITE 5,2,CARR,CARRV
.WAIT 5
JMP NEXT /JUMP OUT OF MACRO.
A .WRITE 5,2,NOTEA,NOTEAV
.WAIT 5
.WRITE 5,2,NOTER,NOTERV
.WAIT 5
B .WRITE 5,2,NOTED,NOTEDV
.WAIT 5
.WRITE 5,2,CONTA,CONTAV
.WAIT 5
.WRITE 5,2,CONTB,CONTBV
.WAIT 5
.READ 4,2,INBUF,4

```



```

        .WAIT 4
        .WRITE 5,2,CARR,CARRV
        .WAIT 5
        LAC INBUF+2
        AND (774000 /MASK
        SAD (434000 /ASCII'G.
        JMP .+2
        JMP B /WRITE NOTED.
        701627 /ISSUE FREE IOTS.
        702212 /READ-IN GATE TRANSFER.
        701627 /TOGGLE FLIP FLOP.
        AND LIT /MASK ALL BUT ERROR BIT.
        SNA
        JMP C /READ-IN ERROR.
        .WRITE 5,2,NOTEV,NOTEV
        .WAIT 5
        .WRITE 5,2,CARR,CARRV
        .WAIT 5
        JMP NEXT /JMP OUT OF MACRO.
C      .WRITE 5,2,NOTEV,NOTEV
        .WAIT 5
        .WRITE 5,2,CARR,CARRV
        .WAIT 5
        .ENDM

/
/CALL BIT MACRO TO TEST EACH BIT AND
/OUTPUT ERROR MESSAGES.
/
        .GEN /GENERATE MACROS.
        BIT (400000,NOTE0,NOTE0V,A1,B1,$
C1, NEXT1
        .NOGEN /SUPPRESS MACRO GENERATION
NEXT1  BIT (200000,NOTE1,NOTE1V,A2,B2,$
C2, NEXT2
NEXT2  BIT (100000,NOTE2,NOTE2V,A3,B3,$
C3, NEXT3
NEXT3  BIT (40000,NOTE3,NOTE3V,A4,B4,$
C4, NEXT4
NEXT4  BIT (20000,NOTE4,NOTE4V,A5,B5,$
C5, NEXT5
NEXT5  BIT (10000,NOTE5,NOTE5V,A6,B6,$
C6, NEXT6
NEXT6  .WRITE 5,2,CARR,CARRV
        .WAIT 5
        .WRITE 5,2,NOTE6,NOTE6V
        .WAIT 5
        JMP* CONREG
ALLOC  .WRITE 5,2,OK,OKV

```

```

      .WAIT 5
      .WRITE 5,2,CARR,CARRV
      .WAIT 5
      JMP* CONREG
/
/I/O REGISTERS.
/
RES      0      /TEST RESULT.
INBUF    .BLOCK 4 /INPUT BUFFER.
OK        OKV/2*1000+502; 0 /HEADER.
          .ASCII /CONTROL REGISTER & RE/
          .ASCII /AD-IN GATES OKAY./<15>
OKV=-OK
NOTEPV   NOTEPV/2*1000+502; 0 /HEADER
          .ASCII /A PROBLEM EXISTS WITH TH/
          .ASCII /E FOLLOWING BITS/<15>
NOTEPV=-NOTEPV
CONT1V   CONT1V/2*10000+502; 0 /HEADER
          .ASCII /IN EITHER CONTROL REGIST/
          .ASCII /ER OR READ-IN/<15>
CONT1V=-CONT1V
CONT2V   CONT2V/2*1000+502; 0 /HEADER
          .ASCII /GATES. AFTER EACH PRINT /
          .ASCII /OUT CHECK THE/<15>
CONT2V=-CONT2V
CONT3V   CONT3V/2*1000+502; 0 /HEADER
          .ASCII /LAMP CORRESPONDING TO TH/
          .ASCII /AT BIT. IF IT IS/<15>
CONT3V=-CONT3V
CONT4V   CONT4V/2*1000+502; 0 /HEADER
          .ASCII /LIT TYPE AN R INDICATING/
          .ASCII / THAT THE READ-/<15>
CONT4V=-CONT4V
CONT5V   CONT5V/2*1000+502; 0 /HEADER
          .ASCII /IN GATE IS IN ERROR. IF /
          .ASCII /LAMP NOT LIT/<15>
CONT5V=-CONT5V
CONT6V   CONT6V/2*1000+502; 0 /HEADER
          .ASCII /TYPE C INDICATING ERROR /
          .ASCII /IN CONTROL/<15>
CONT6V=-CONT6V
CONT7V   CONT7V/2*1000+502; 0 /HEADER
          .ASCII /REGISTER AND POSSIBLE ER/
          .ASCII /ROR IN/<15>
CONT7V=-CONT7V
CONT8V   CONT8V/2*1000+502; 0 /HEADER
          .ASCII /REAL-IN GATE./<15>
CONT8V=-CONT8V

```

```

NOTE0 NOTE0V/2*1000+502; 0 /HEADER
      .ASCII /ERROR IN BIT 0/<15>
NOTE0V=-NOTE0
NOTE1 NOTE1V/2*1000+502; 0 /HEADER
      .ASCII /ERROR IN BIT 1/<15>
NOTE1V=-NOTE1
NOTE2 NOTE2V/2*1000+502; 0 /HEADER
      .ASCII /ERROR IN BIT 2/<15>
NOTE2V=-NOTE2
NOTE3 NOTE3V/2*1000+502; 0 /HEADER
      .ASCII /ERROR IN BIT 3/<15>
NOTE3V=-NOTE3
NOTE4 NOTE4V/2*1000+502; 0 /HEADER
      .ASCII /ERROR IN BIT 4/<15>
NOTE4V=-NOTE4
NOTE5 NOTE5V/2*1000+502; 0 /HEADER
      .ASCII /ERROR IN BIT 5/<15>
NOTE5V=-NOTE5
NOTEX NOTEXV/2*1000+502; 0 /HEADER
      .ASCII /TYPE R OR C - /<175>
NOTEXV=-NOTEX
CARR CARRV/2*1000+502; 0 /HEADER
      .ASCII <15>
CARRV=-CARR
NOTER NOTERV/2*1000+502; 0 /HEADER
      .ASCII /READ-IN GATE ERROR IN AB/
      .ASCII /OVE BIT./<15>
NOTERV=-NOTER
NOTE6 NOTE6V/2*1000+502; 0 /HEADER
      .ASCII /CONTROL REGISTER ERROR A/
      .ASCII /ND POSSIBLY/<15>
NOTE6V=-NOTE6
NOTED NOTEDV/2*1000+502; 0 /HEADER
      .ASCII /PATCH A WORKING IOT TO J/
      .ASCII /&K INPUTS OF A/<15>
NOTEDV=-NOTED
CONTA CONTA/2*1000+502; 0 /HEADER
      .ASCII /FLIP FLOP AND ITS OUTPUT/
      .ASCII / TO THE ERROR/<15>
CONTA/=-CONTA
CONTB CONTBV/2*1000+502; 0 /HEADER
      .ASCII /BIT IN THE READ-IN GATE,/
      .ASCII / THEN TYPE G./<15>
CONTBV=-CONTB
NOTE7 NOTE7V/2*1000+502; 0 /HEADER
      .ASCII /CONTROL REGISTER ERROR /
      .ASCII /ONLY./<15>
NOTE7V=-NOTE7

```

```
NOTEF NOTEFV/2*1000+502; 0 /HEADER
      .ASCII /CONTROL REGISTER & READ-/
      .ASCII /IN GATE ERROR./<15>
NOTEFV=-NOTEF
NOTEG NOTEGV/2*1000+502; 0 /HEADER
      .ASCII /NO FUTHER ERRORS./<15>
NOTEGV=-NOTEG
      .END
```

```

.TITLE PRECTR /MACRO SUBROUTINE
    /CALLED BY LOCUT1 TO
    /TEST PRESET COUNTERS.
.GLOBL PRECTR, ERROR
    .IODEV 5
PRECTR 0 /ENTRY POINT.
    .INIT 5,1
    LAC 0 /CLEAR CONTROL REG. FOR
    701647 /MDAC/ADC TEST; NEXT.
    LAC (-1130 /DECIMAL 600.
    DAC INCR
    LAC (1127 /DECIMAL 599.
    ALS 10 /YIELDS 1127 IN AC 0-9.
    CMA
    701546 /CLEAR & LOAD CTR1.
IIOTA 701621 /ISSUE IOT1.
    NOP
    NOP
    701541 /SKIP ON FREE FLAG 0.
    JMP .+2
    JMP OUT1
    ISZ INCR
    JMP IIOTA
    JMP SET1 /CTR FAILS TO SET ON
    /600 PULSES.
OUT1 701624 /RESET FLIP FLOP WITH IOT4
    LAC INCR /INCR HASN'T BEEN
    TAD (1 /INCREMENTED ON LAST
    /COUNT.
    DAC ERRI
    SZA
    JMP CTR1 /CTR1 COUNTS WRONG.
    .WRITE 5,2, NOTEA, NOTEAV /CTR1 OK.
    .WAIT 5
BC2 701620 /BEGIN CTR2 TEST.
    LAC (-1130 /BEGIN CTR2 TEST.
    DAC INCR
    LAC (1127
    ALS 10
    CMA
    701526
IIOTB 701622 /ISSUE IOT2.
    NOP
    NOP
    701521 /SKIP ON FREE FLAG 1.
    JMP .+2
    JMP OUT2
    ISZ INCR
    JMP IIOTB

```

```

      JMP SET2
OUT2  701624 /RESET FLIP FLOP WITH IOT4
      LAC INCR
      TAD (1
      DAC ERR2
      SZA
      JMP CTR2
      .WRITE 5,2,NOTE2,NOTE2V /CTR2 OK.
      .WAIT 5
      JMP* PRECTR
CTR1  JMS* ERROR
      JMP .+3
      PC1
      ERRI#
      JMP BC2
SET1  .WRITE 5,2,NOTE2,NOTE2V
      .WAIT 5
      JMP BC2
CTR2  JMS* ERROR
      JMP .+3
      PC2
      ERR2#
      JMP* PRECTR
SET2  .WRITE 5,2,NOTE4,NOTE4V
      .WAIT 5
      JMP* PRECTR
PC1   1
PC2   2
NOTEA NOTEAV/2*1000+502; 0
      .ASCII /PRE-SET COUNTER 1 /
      .ASCII /OKAY./<15>
NOTEAV=-NOTEA
NOTE2 NOTE2V/2*1000+502; 0
      .ASCII /PRE-SET COUNTER 2 /
      .ASCII /OKAY./<15>
NOTE2V=-NOTE2
NOTE2 NOTE2V/2*1000+502; 0
      .ASCII /COUNTER 1 FAILS TO SET/
      .ASCII / AFTER 600 PULSES./<15>
NOTE2V=-NOTE2
NOTE4 NOTE4V/2*1000+502; 0
      .ASCII /COUNTER 2 FAILS TO SET/
      .ASCII / AFTER 600 PULSES./<15>
NOTE4V=-NOTE4
      .END

```

ERROR

PAGE 1

```
      SUBROUTINE ERROR (ICTR,IERR)
C OUTPUT FOR PRECTR OF LOCUT1.
C NOTE: IERR IS NEGATIVE.
      ICOUNT = 600 + IERR
      WRITE(5,1) ICTR,ICOUNT
1     FORMAT(* COUNTER *,11,* SETS AFTE
      #R COUNTING TO *,13,*,*)
      RETURN
      END
```

```

        .TITLE CONV      /SUBROUTINE CALLED
        .GLOBL CONV, .DA /BY LOCUT1 TO
CONV    0                /TEST MDAC'S
        JMS* .DA          /AND ADC'S.
        JMP .+3

```

```

MDAC    0
ADC     0

```

```

/
/CLEAR CONTROL REGISTER.
LAC (0
701647

```

```

/
/CHOOSE ADC AND MDAC.
/

```

```

CKDAC   LAC* MDAC
        SAD (1
        JMP M1
        SAD (2
        JMP M2
        SAD (3
        JMP M3
        SAD (4
        JMP M4
        JMP* CONV

```

```

CKADC   LAC* ADC
        SAD (1
        JMP A1
        SAD (2
        JMP A2
        SAD (3
        JMP A3
        SAD (4
        JMP M4
        JMP* CONV

```

```

/
/THE VARIABLE REG WILL BE USED WITH THE
/CONTROL REGISTER TO PUT LOCUST IN RESET
/AND COMPUTE (USING BIT 0) AND TO CON-
/TROL THE COMMAND SIGNALS FOR THE ADC'S.
/CONTROL REG. BITS 1-4 ARE PATCHED TO
/NOR GATES ALONG WITH THE COMPLEMENT OF
/THE R PULSE. THE GATE OUTPUT PROVIDES
/THE COMMAND FOR THE ADC'S WHEN THE COR-
/RESPONDING CONTROL REG. BIT IS 0.

```

```

/
/TEST THE ADC'S AND MDAC'S IN PAIRS AS
/DESIGNATED BY LOCUT1 GFROM TTY INPUT.
/

```



```

I  LAS.
31 0 /CLEAR, LOAD, AND TRANSFER
    /TO MDAC.
    LAC REG
    701647
SKI 0 /SKIP ON ADC FLAG.
    JMP , -1
R1 0 /READ ADC.
    AND (777600 /MASK OFF UNUSED
        /BITS.
    HLT
    LAC (360000 /PUT LOCUST IN RESET.
    701647
    LAS
    SAD (1
    JMP* CONV
    JMP CONT

```

INITIALIZATION OF INSTRUCTIONS.

```

LAC (703427
DAC MDACI
JMP CKADC
LAC (703527
DAC MDACI
JMP CKADC
LAC (703627
DAC MDACI
JMP CKADC
LAC (703727
DAC MDACI
JMP CKADC
LAC (703441
DAC ADCSKI
LAC (703452
DAC ADCRI
LAC (560000
DAC REG#
JMP CONT
LAC (703541
DAC ADCSKI
LAC (703552
DAC ADCRI
LAC (660000
DAC REG
JMP CONT
LAC (703641
DAC ADCSKI

```

CONV

PAGE 3

LAC (703652
DAC ADCRI
LAC (720000
DAC REG
JMP CONT
LAC (703741
DAC ADCSKI
LAC (703652
DAC ADCRI
LAC (740000
DAC REG
JMP CONT
.END